link alle slide

OpenSCAD

Tutorial 1

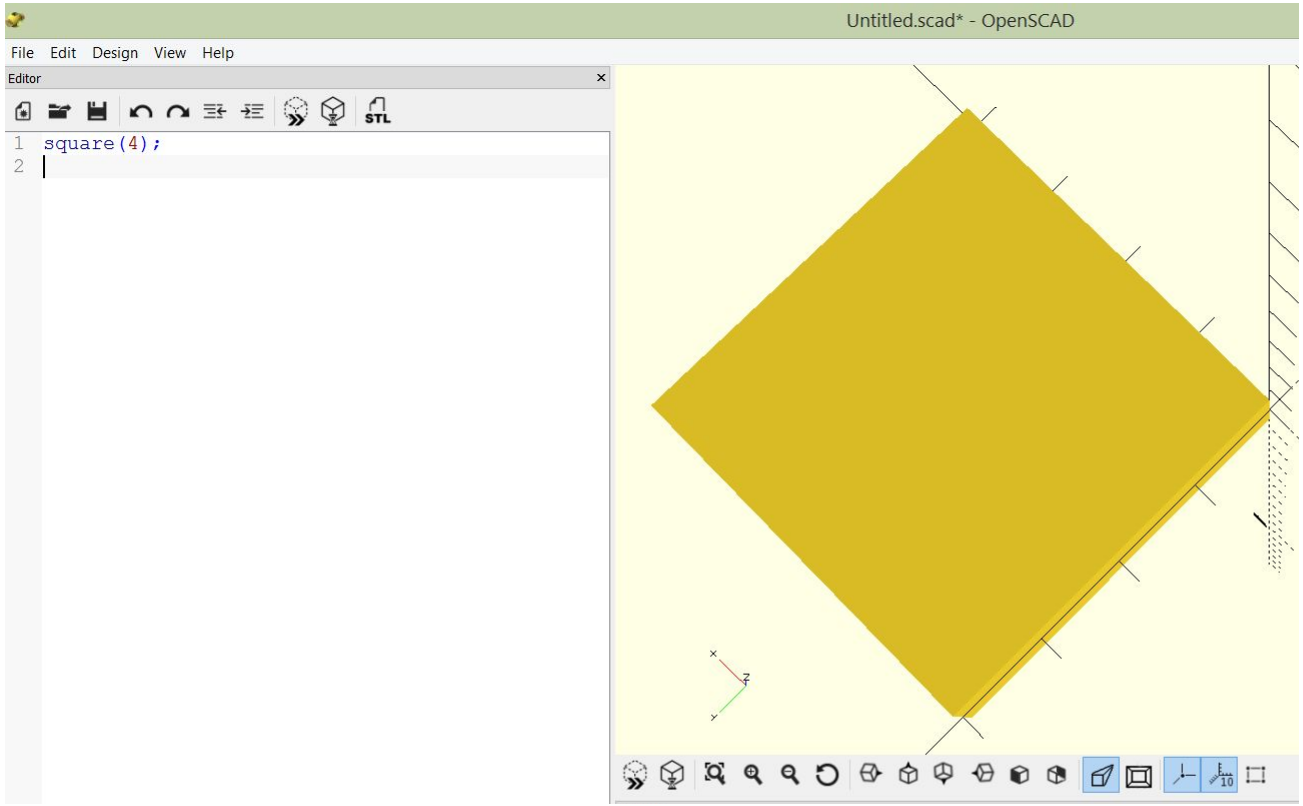
2D Shapes

```
circle(radius | d=diameter)
square(size,center)
square([width,height],center)
polygon([points])
polygon([points],[paths])
text(t, size, font, halign, valign,
spacing, direction, language, script)
```

2D Shapes

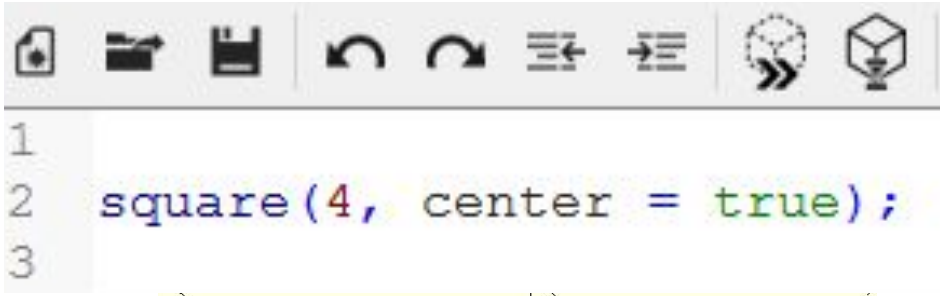
Create some 2D shapes. Try

```
circle(...);
square(...);
```

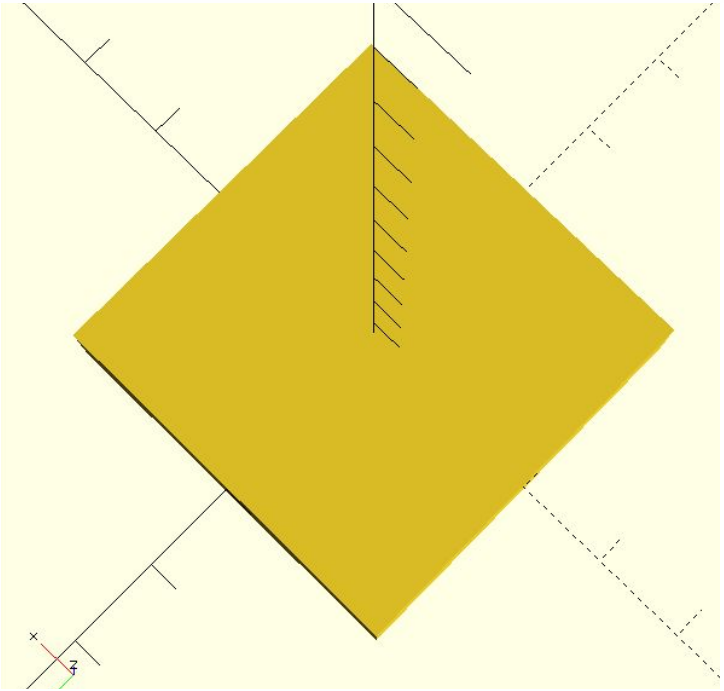


Square

Center the square on the origin.



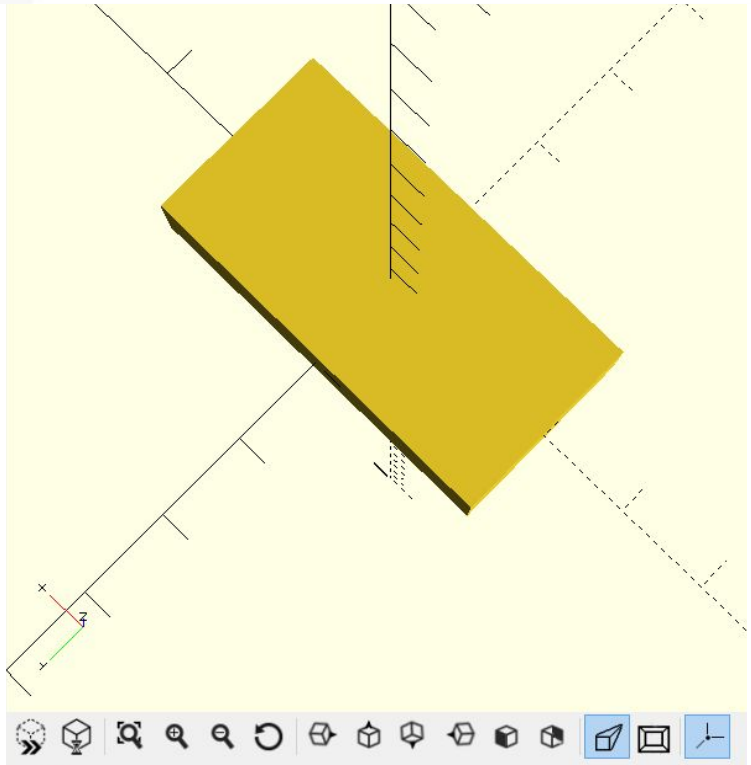
```
1
2  square(4, center = true);
3
```



Rectangle

Create a rectangle and center it on the origin.

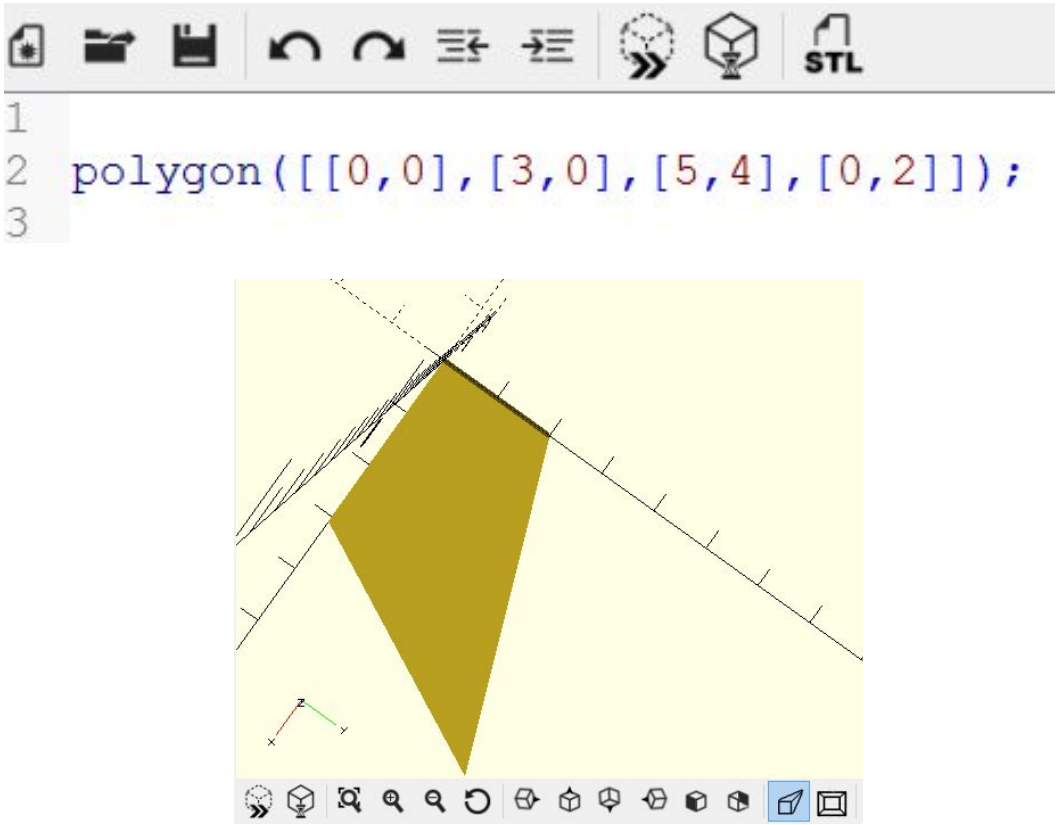
```
1
2  square([4, 2], center = true);
3
```



Polygon

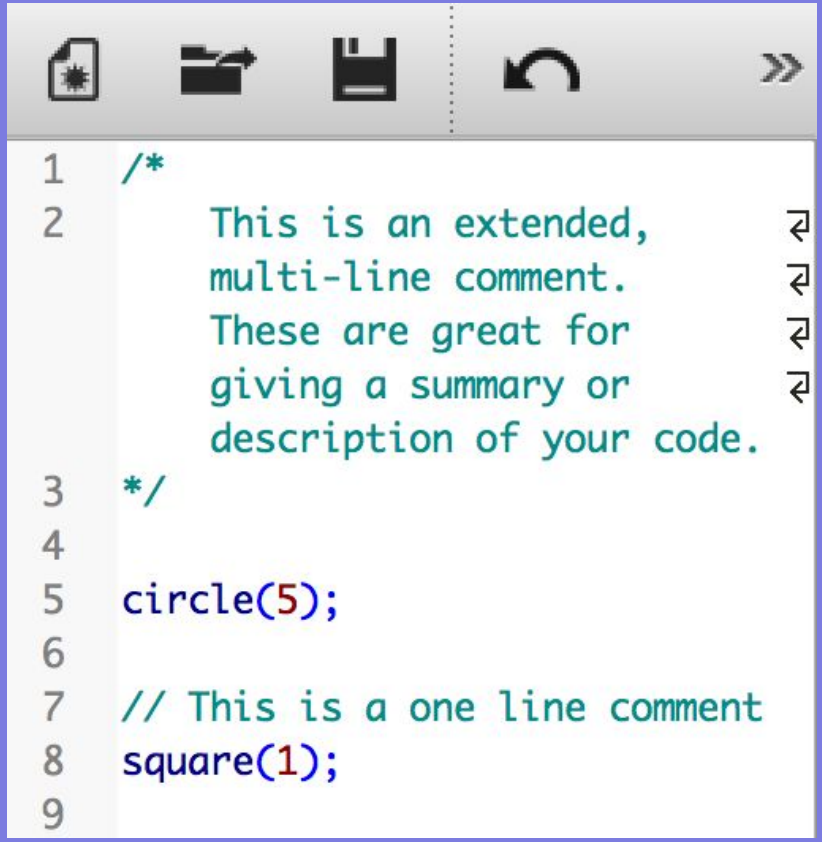
Create a polygon. The points must be in the order that you would like to draw the polygon; otherwise, provide a path.

```
polygon([points…]);
     or
polygon([points…],
          [paths...]);
```



Note about Comments

- **Comments make code readable.**
- There are two ways to make comments.



```
/*
    This is an extended,
    multi-line comment.
    These are great for
    giving a summary or
    description of your code.
*/

circle(5);

// This is a one line comment
square(1);
```

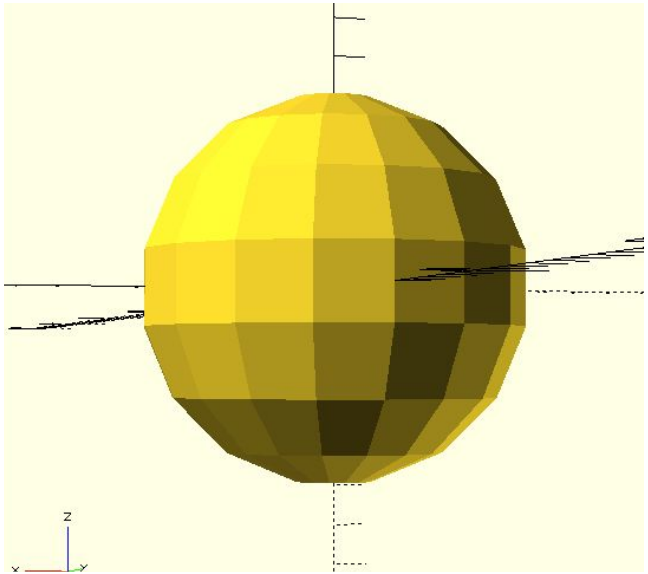
3D Shapes

```
sphere(radius | d=diameter)
cube(size, center)
cube([width,depth,height], center)
cylinder(h,r|d,center)
cylinder(h,r1|d1,r2|d2,center)
polyhedron(points, triangles, convexity)
```

Sphere

Create a sphere.

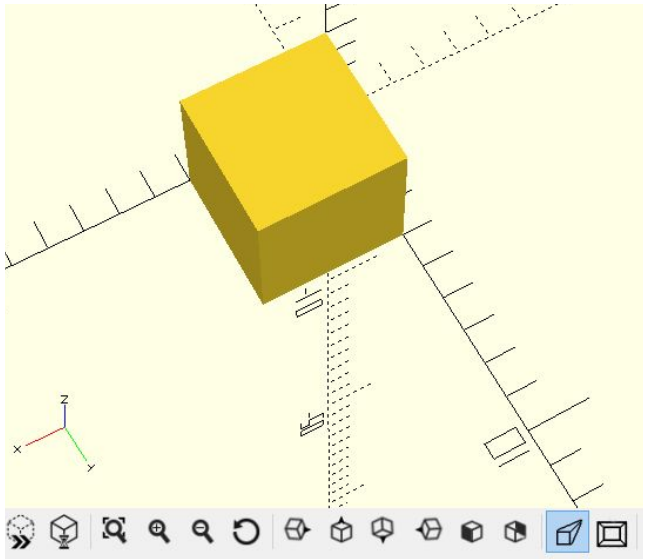
```
1
2  // all of these create the same
       sphere
3  sphere(5);
4  sphere(r=5);
5  sphere(d=10);
```



Cube

Create a cube.

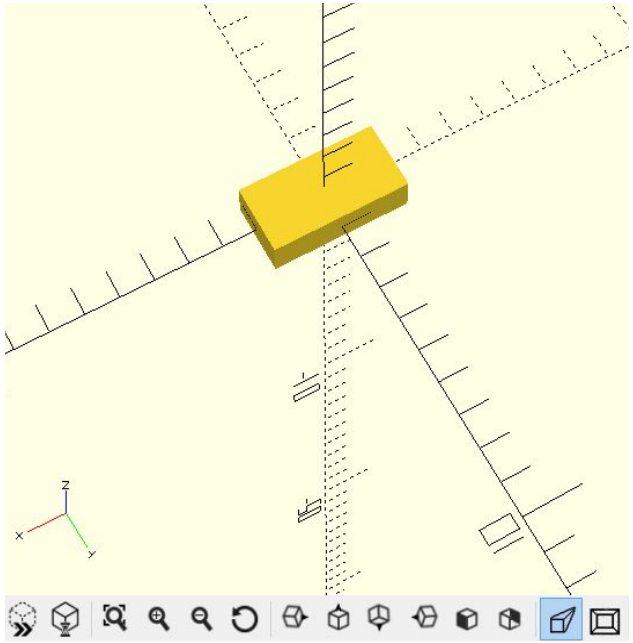
```
1
2  cube(4);
3
```



BOX

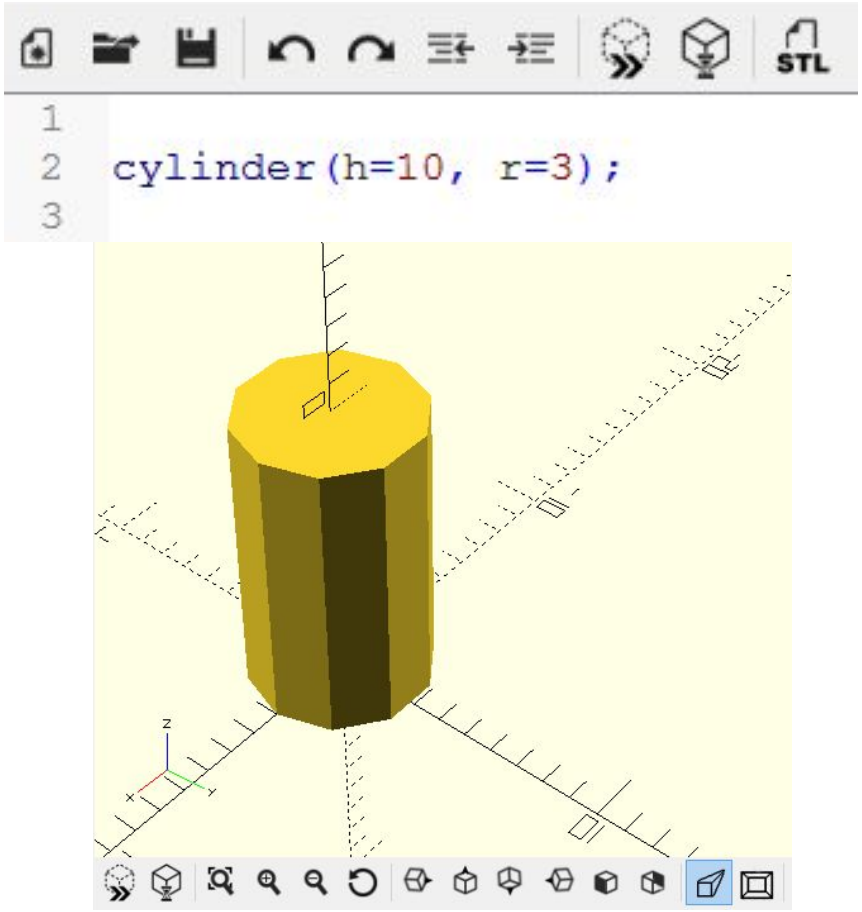
Create a rectangular cube
and center it on the origin.

```
1
2  cube([4,2,1], center = true);
3
```



Cylinder

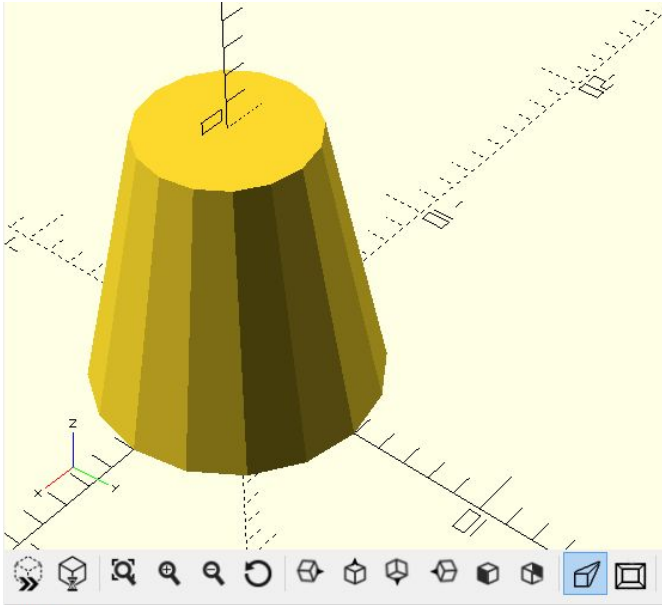
Create a cylinder.



Cone

Create a cone.

```
1
2  cylinder(h=10, r1=5, r2=3);
3
```



Note about Debugging

- Debugging tools **make troubleshooting faster.**

```
*disable
!show only
#highlight / debug
%transparent / background
```

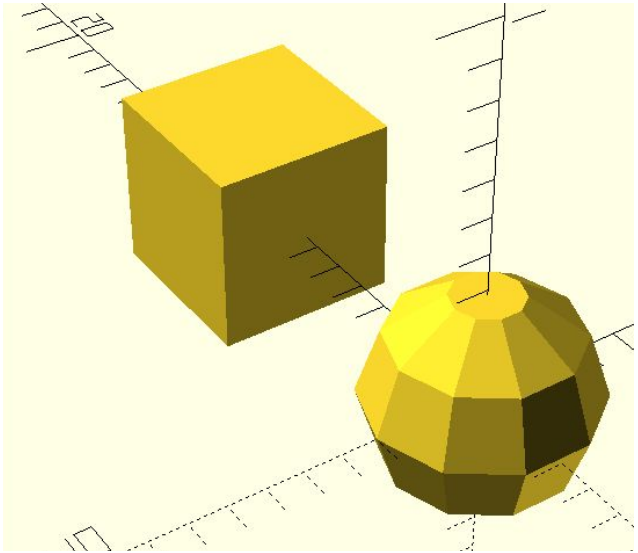
Transformations

```
translate([x,y,z])
rotate([x,y,z])
scale([x,y,z])
resize([x,y,z],auto)
mirror([x,y,z])
multmatrix(m)
color("colorname")
color([r,g,b,a])
offset(r|delta,chamfer)
hull()
minkowski()
```

Translate

Use translate to move the cube on
the y axis.

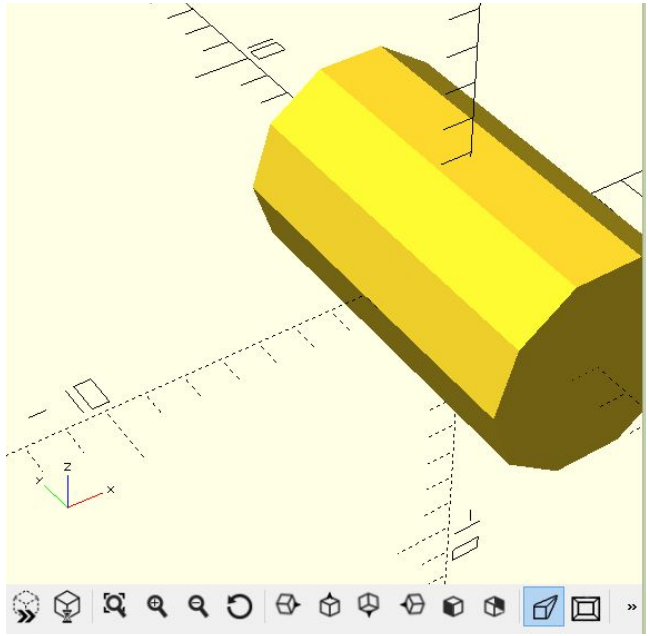
```
1
2 sphere(3);
3
4 translate([0,10,0]) {
5     cube(5, center=true);
6 }
```



Rotate

Rotate a cylinder around the x axis.

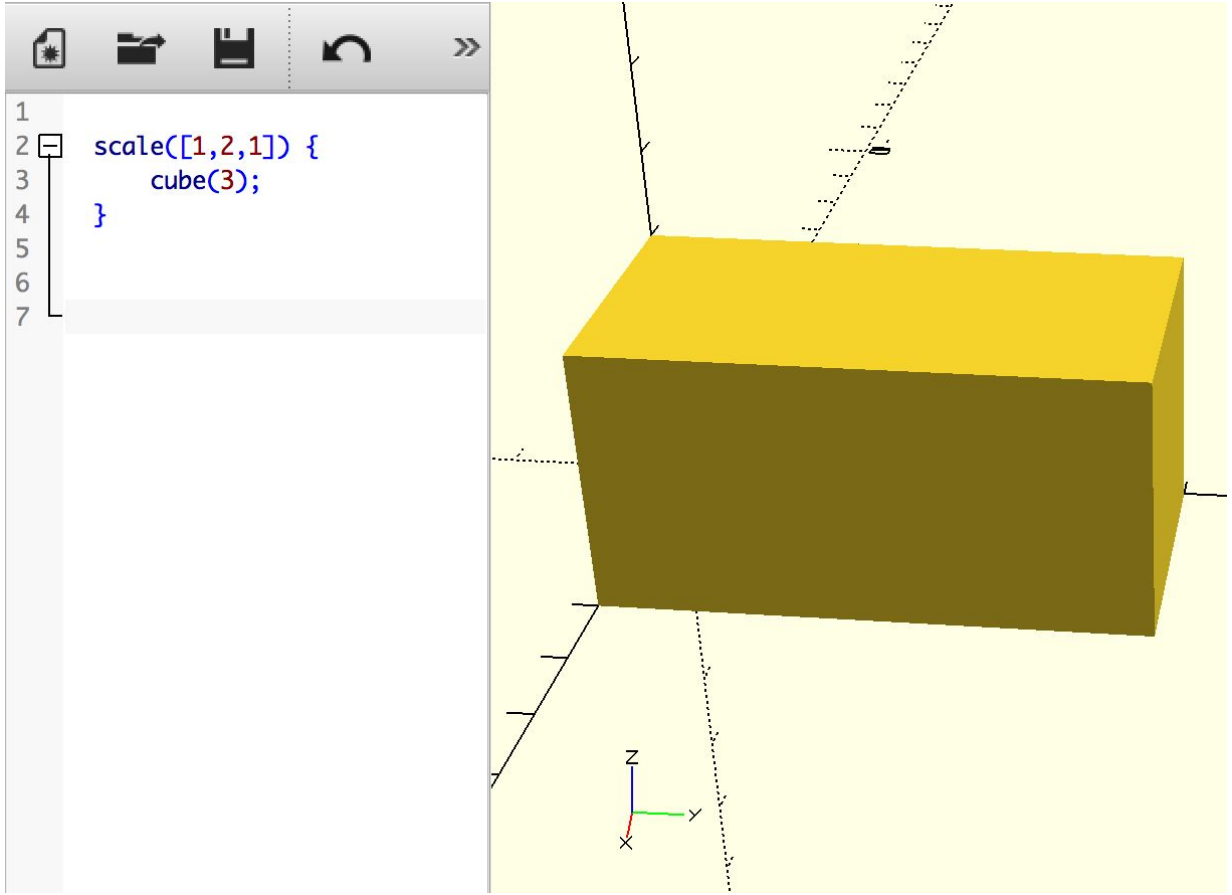
```
rotate([90,0,0]) {
    cylinder(r=3, h=10,
    center=true);
}
```



Scale

Scale takes a value for x, y, and z (1 = 100%, 2=200%, etc.).

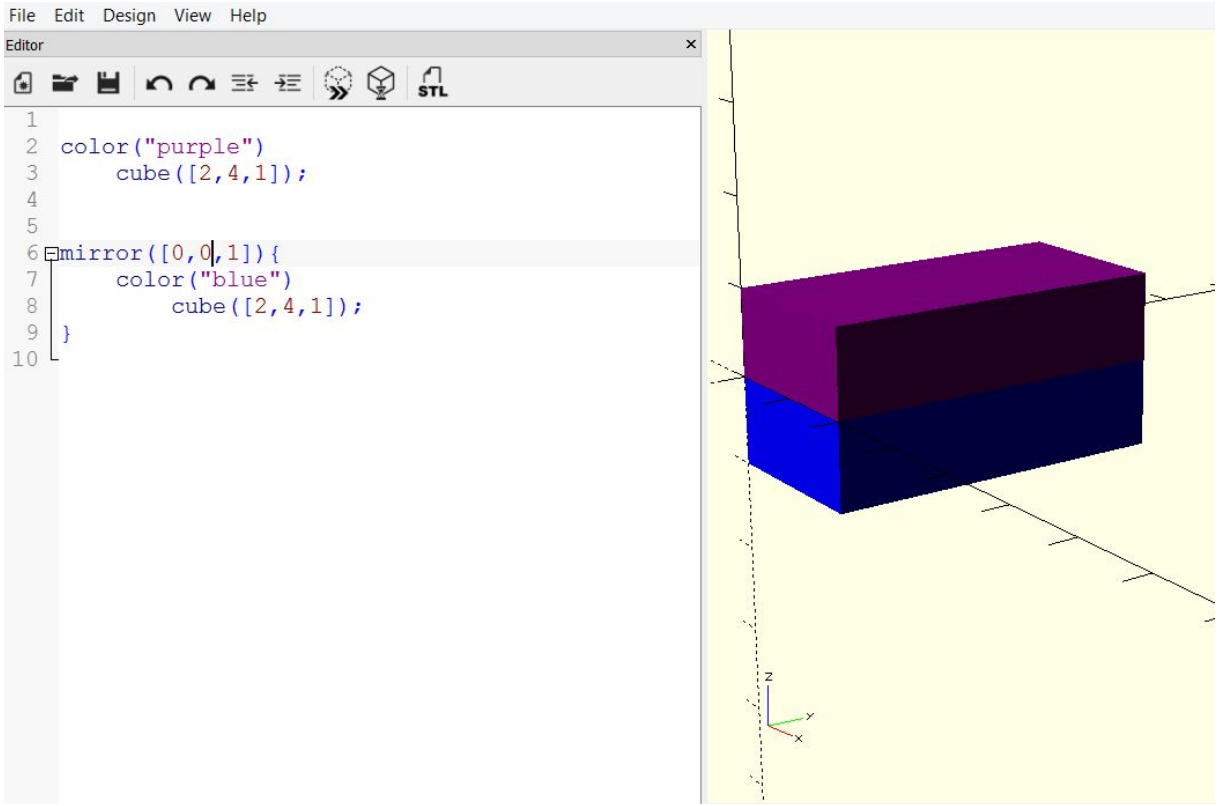
Create an ellipse.



Mirror

Mirror the sphere about the origin. Set x, y, and z equal to 1 or 0 depending upon which plane you want to mirror.

More info here.



Color

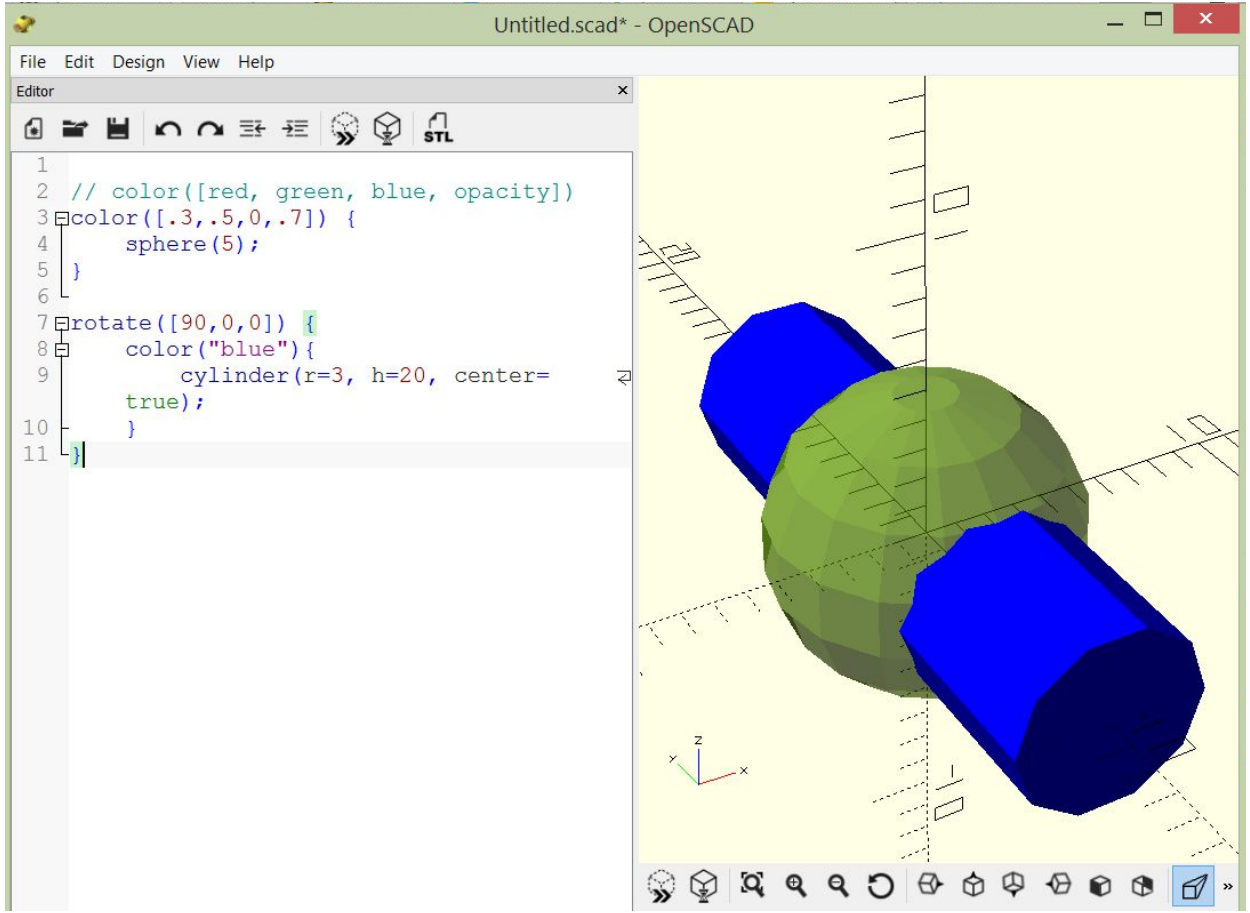
There are two ways to use color:

color([red, green, blue, alpha])

or

color(“*name of color*”)

Change the color of objects.

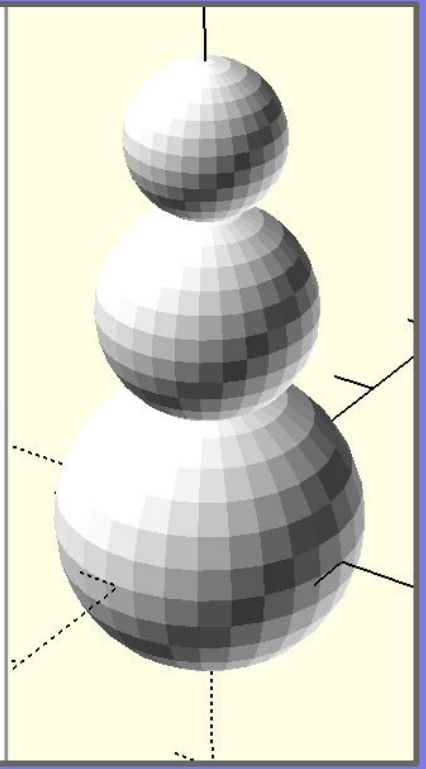


NOTE ABOUT SYNTAX

- **use {}** if the command applies to more than one object/line

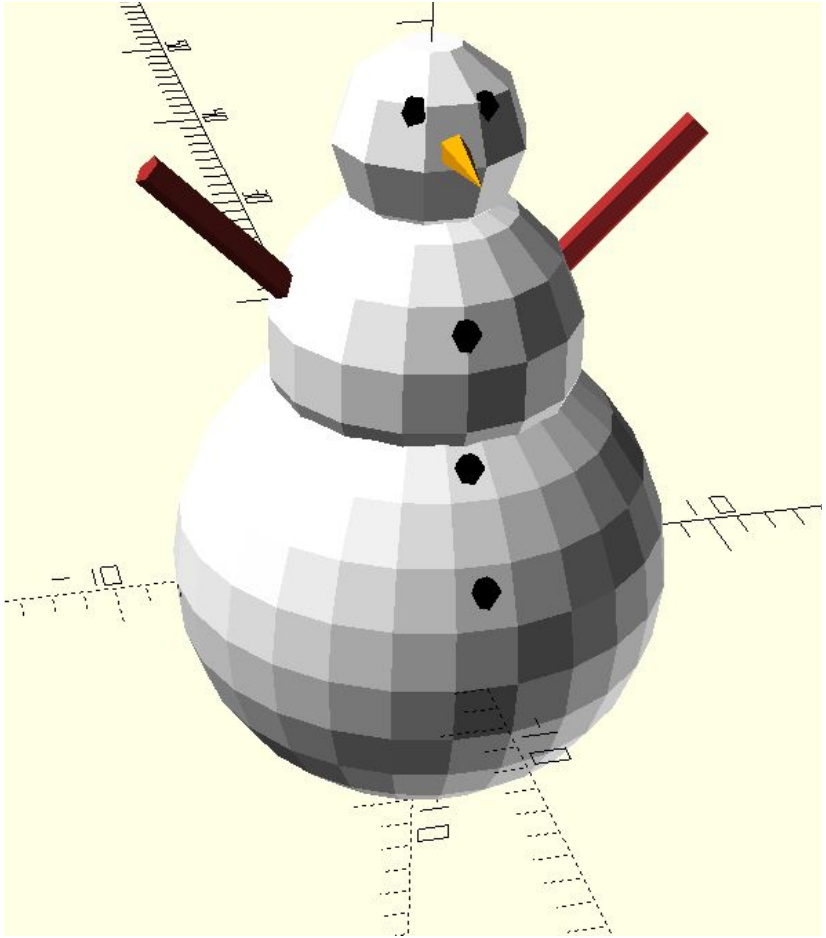
- **Indent lines** to make code readable

```
2   color("white")
3   sphere(r=100);
4   color("white")
5   translate([0,0,150])
6   sphere(r=70);
7   color("white")
8   translate([0,0,265])
9   sphere(r=50);
10
11  // same code
12  color("white") {
13      sphere(r=100);
14      translate([0,0,150])
15          sphere(r=70);
16      translate([0,0,265])
17          sphere(r=50);
18  }
```



EXERCISE

Build a snowman



OpenSCAD

Tutorial 2

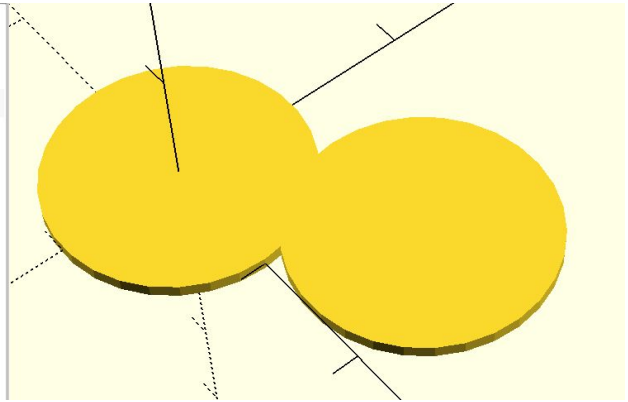
Transformations

```
translate([x,y,z])
rotate([x,y,z])
scale([x,y,z])
resize([x,y,z],auto)
mirror([x,y,z])
multmatrix(m)
color("colorname")
color([r,g,b,a])
offset(r|delta,chamfer)
hull()
minkowski()
```

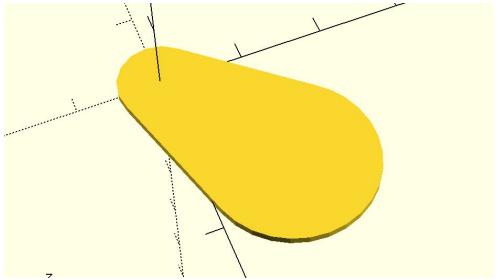
HULL

Join two shapes together by filling in the space around their collective outline.

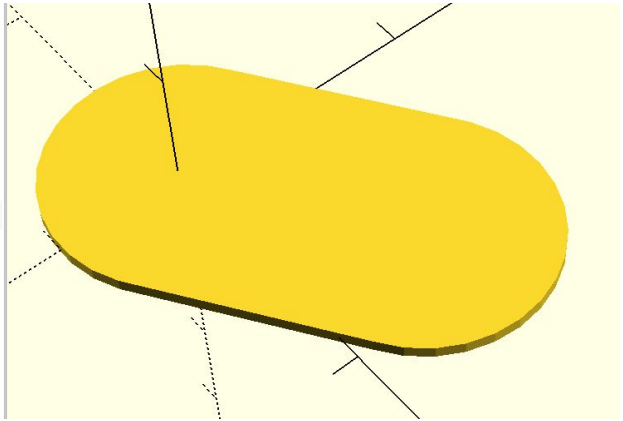
```
translate([15,10,0]) circle(10);
circle(10);
```



Try making a pear shape.



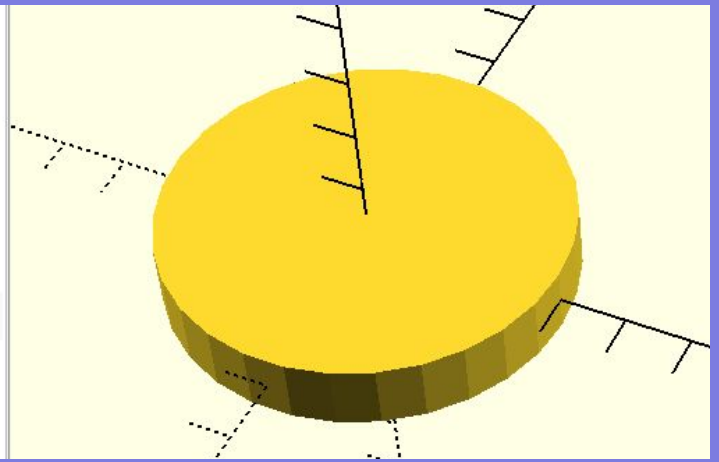
```
hull() {
    translate([15,10,0]) circle(10);
    circle(10);
}
```



NOTE $FN - NUMBER OF FRAGMENTS

Increase $fn to increase the smoothness (number of fragments) of circles.

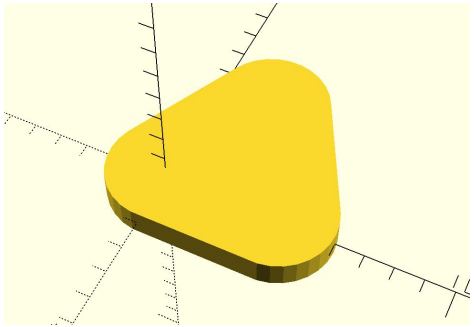
```
2
3   $fn=30;
4
5   circle(3);
6
7
8
```



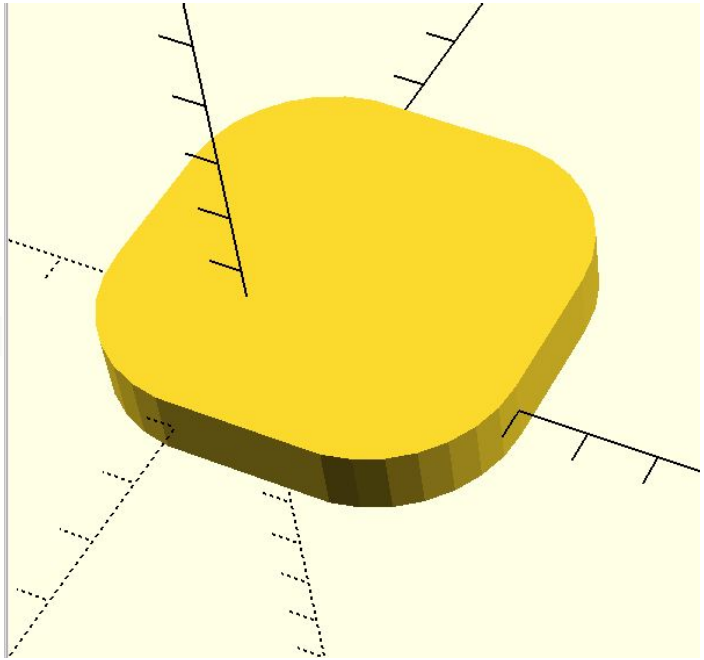
Minkowski

Minkowski() is good for creating rounded edges along shapes. It moves the second shape around the outline of the first shape.

Create a rounded triangle.



```
1
2  $fn=30;
3
4  minkowski() {
5      square(2);
6      circle(2);
7  }
8
9
```



Modules

Modules are blueprints for code objects. They prevent code repetition, and keep your code readable.

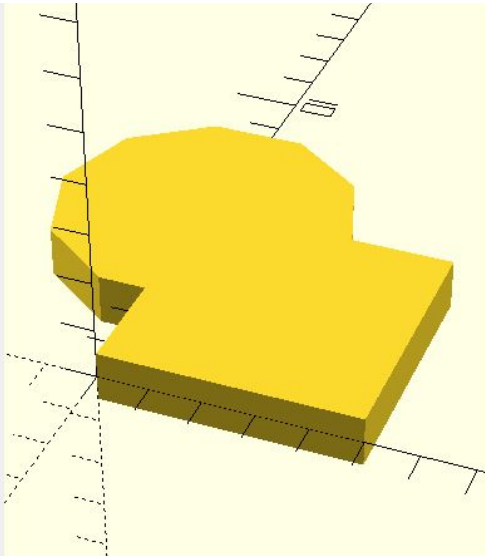
Module

Use a module() to encapsulate a series of commands. Call the module to use it!

Create your own module and call it in your program.

Editor

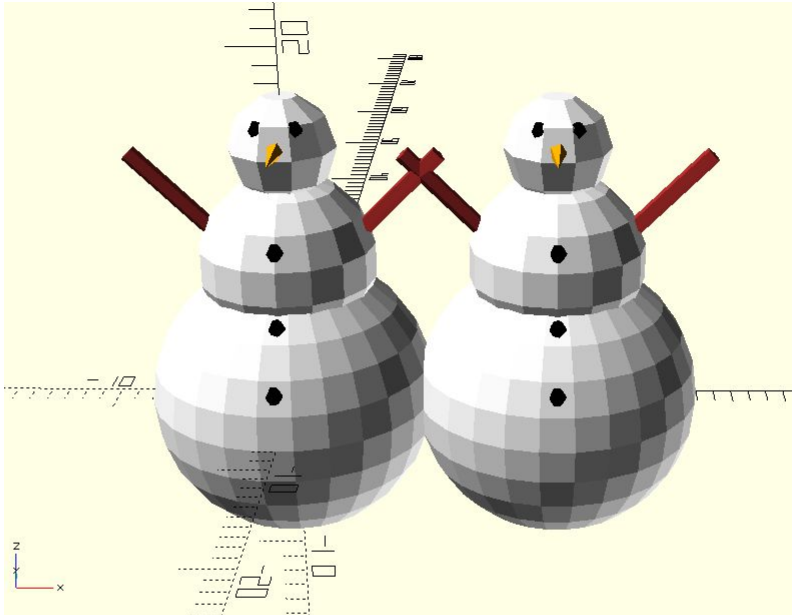
```
1
2  // create module
3  module myModule() {
4      square(5);
5      translate([0,5,0]) {
6          circle(3);
7      }
8  }
9
10 // call module
11 myModule();
12
13
```



EXERCISE

Build 2 snowmen

Use a module() to easily duplicate your snowman!



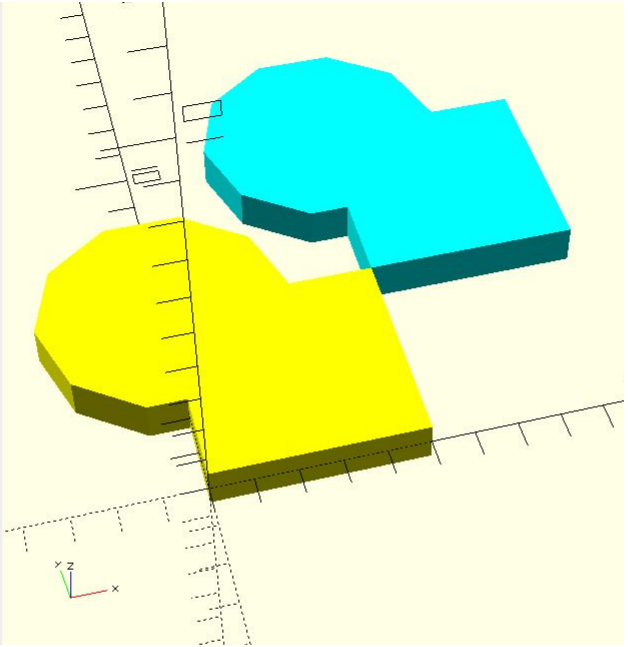
Passing Arguments

It's possible to pass values to modules, which is useful if you'd like to prevent code repetition.

In this example, we're passing x, y, and color coordinates so that we don't have to call translate() to move the module or color() to change its color.

Try creating your own module and passing arguments.

```
1
2  // create module
3  module myModule(xvar, yvar, c) {
4      translate([xvar, yvar, 0]) {
5          color(c){
6              square(5);
7              translate([0,5,0]) {
8                  circle(3);
9              }
10         }
11     }
12 }
13
14 // call module
15 myModule(0,0,"yellow");
16 myModule(5,5,"cyan");
17
18
```

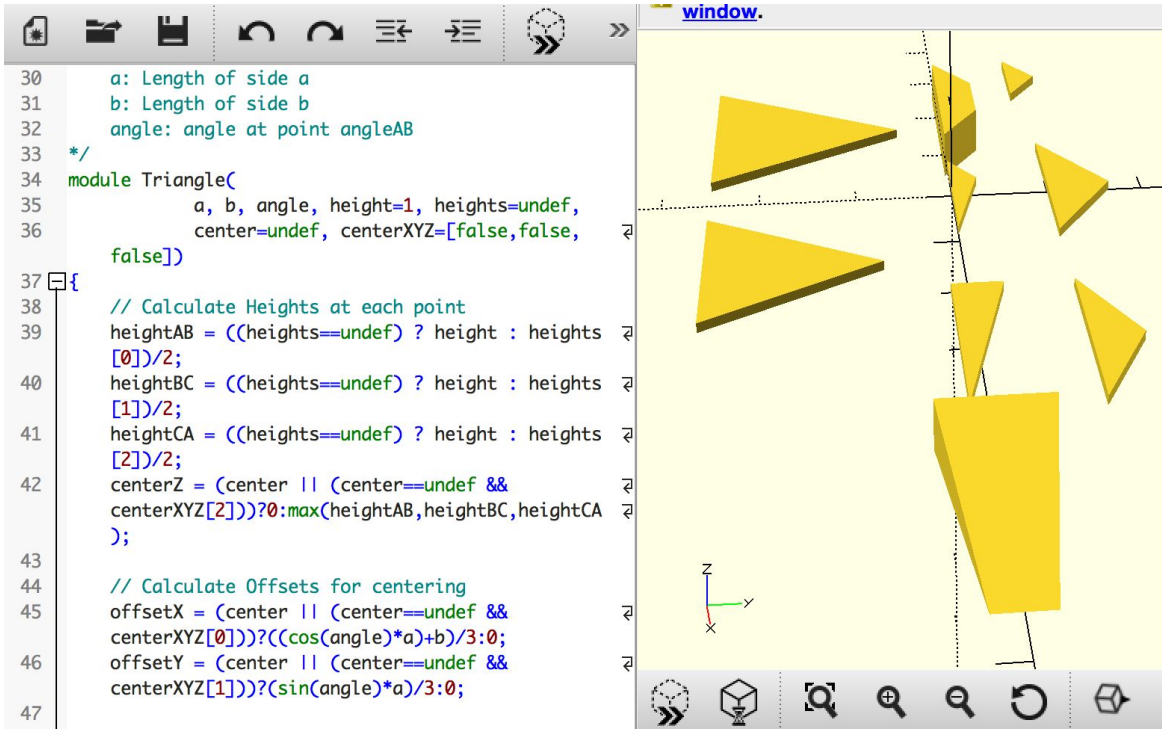


On the Web

Modules on the web can be copied and used in your own code. Make sure to give credit!

This file created modules for easily making triangles.

http://www.thingiverse.com/thing:220868/#files



Iteration

```
for (i = [start:end]) { … }
for (i = [start:step:end]) { … }
```

FOR

For() is a way to repeat a set of steps for a certain number of times- specified by the start and end values.

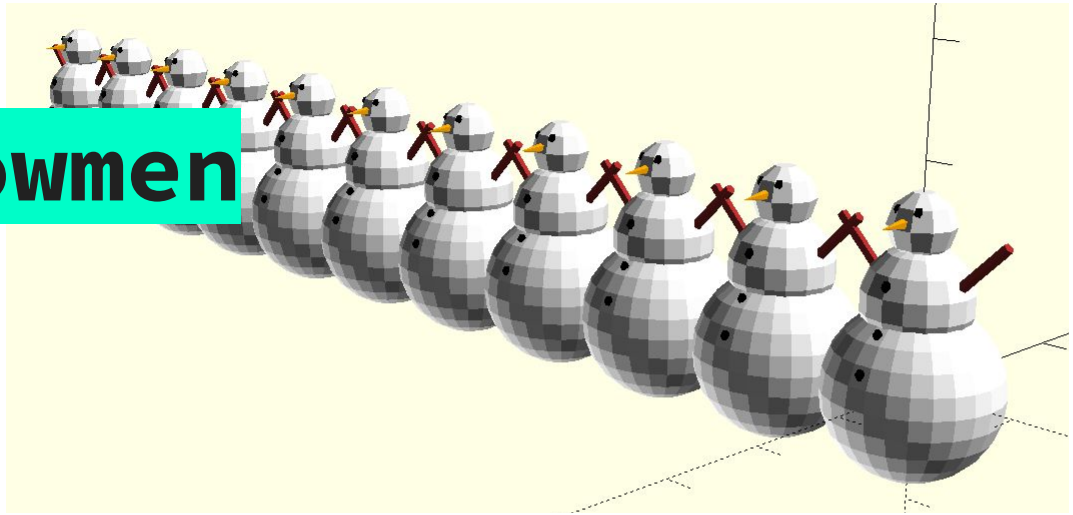
for (i = [start:end]) { … }

```
2
3
4  // make 4 identical circles
5  for(i = [0:4]) {
6      translate([i*4,0,0]) {
7          circle(2);
8      }
9  }
10
```

EXERCISE

Build 10 snowmen

Use a module() and for() to quickly make 10 snowmen.

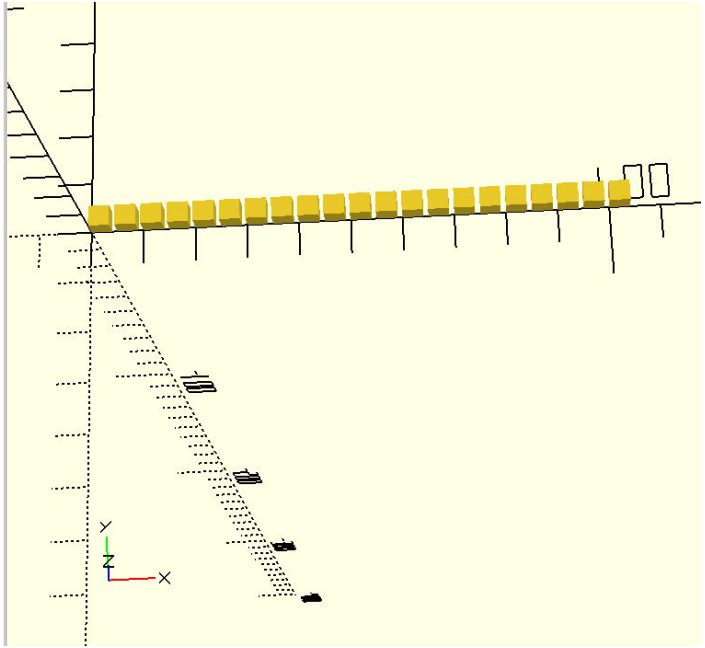


FOR

For() can also take a step interval. Each time the loop repeats, “i” (or the iteration variable) is increased by the value of the step. The loop stops when i gets to 100.

```
for (i = [start:step:end]) {
… }
```

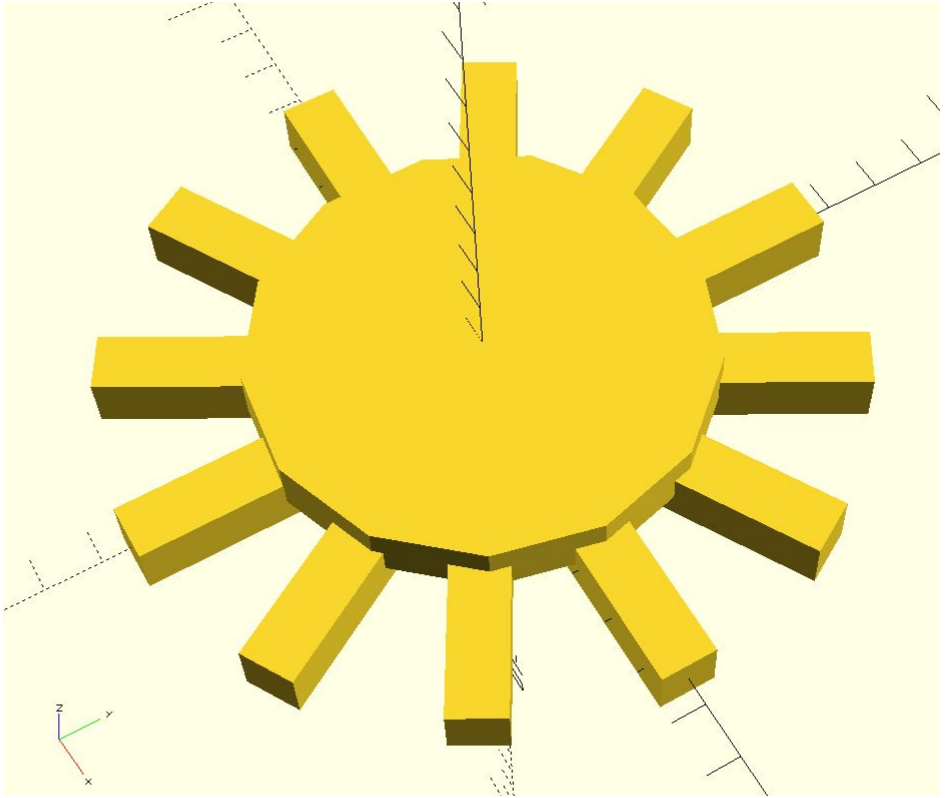
```
3
4  // translate by 5 on the x axis
5  for(i = [0:5:100]) {
6      translate([i,0,0]) {
7          cube(4);
8      }
9  }
10
```



EXERCISE

Build a sun

Use a for() and rotate() to easily make a sun.



2D to 3D

linear_extrude(height,center,convexity,twist,slices)
rotate_extrude(convexity)

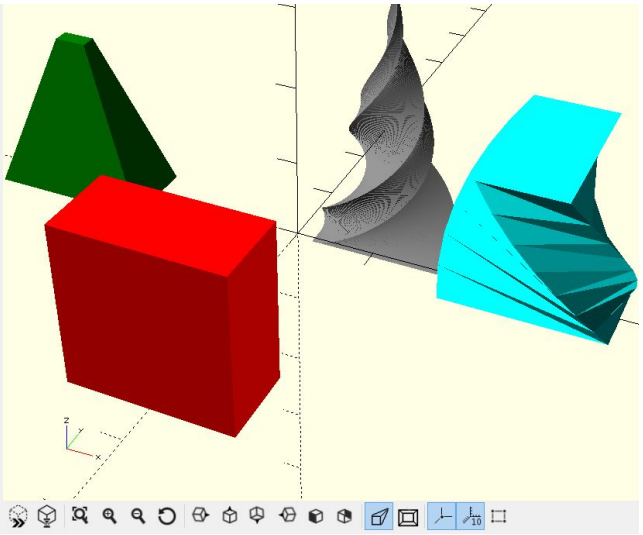
Linear Extrude

linear_extrude() gives a 2D shape some depth. Check out the example file:

File > Examples > Basics > linear_extrude.scad

Use linear_extrude() to make cheese block.

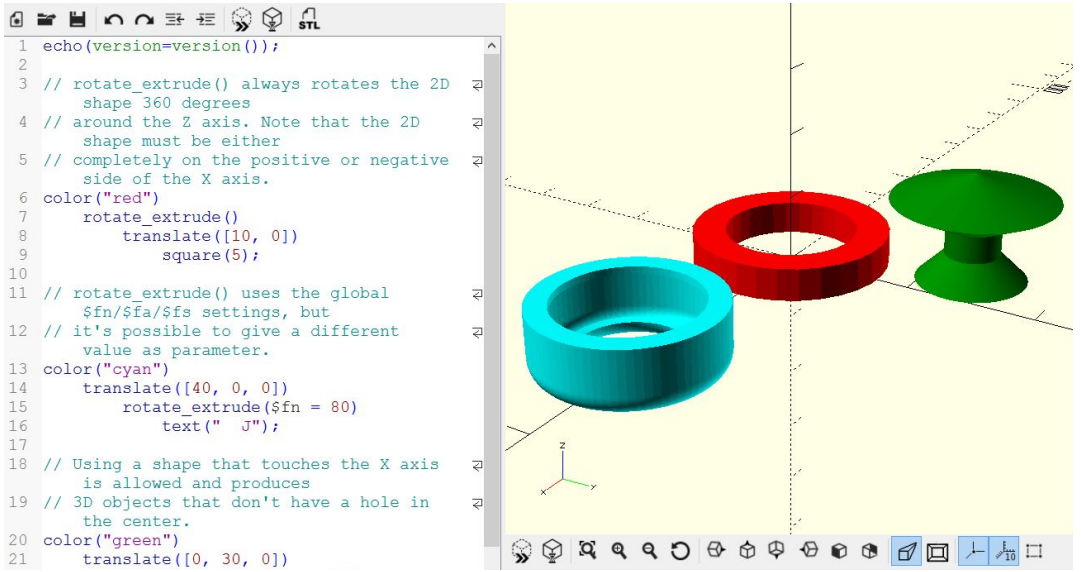
```
echo(version=version());

// simple 2D -> 3D extrusion of a rectangle
color("red")
    translate([0, -30, 0])
        linear_extrude(height = 20)
            square([20, 10], center = true);

// using the scale parameter a frustum can be constructed
color("green")
    translate([-30, 0, 0])
        linear_extrude(height = 20, scale = 0.2)
            square([20, 10], center = true);

// with twist the extruded shape will rotate around the Z axis
color("cyan")
    translate([30, 0, 0])
        linear_extrude(height = 20, twist = 90)
            square([20, 10], center = true);

// combining both relatively complex shapes can be created
color("gray")
    translate([0, 30, 0])
        linear_extrude(height = 40, twist = -360, scale = 0, center = true, slices =
```



Rotate Extrude

rotate_extrude() revolves a 2D shape 360 degrees. Check out the example file:

File > Examples > Basics > rotate_extrude.scad



```
echo(version=version());

// rotate_extrude() always rotates the 2D shape 360 degrees
// around the Z axis. Note that the 2D shape must be either
// completely on the positive or negative side of the X axis.
color("red")
    rotate_extrude()
        translate([10, 0])
            square(5);

// rotate_extrude() uses the global $fn/$fa/$fs settings, but
// it's possible to give a different value as parameter.
color("cyan")
    translate([40, 0, 0])
        rotate_extrude($fn = 80)
            text("  J");

// Using a shape that touches the X axis is allowed and produces
// 3D objects that don't have a hole in the center.
color("green")
    translate([0, 30, 0])
```

EXERCISE

Make a Martini

Use a polygon and rotate_extrude() to make a Martini glass.

(you might want some graph paper…)